IN THE CLAIMS

Please enter amendments as indicated below:

1. (currently amended) A method of identifying a global breakpoint for debugging computer software, said method including the steps of:

receiving a file name for an executable image file, wherein the executable image file is loaded in memory of a computer system and the global breakpoint is to be placed in the image for executing by the computer system;

receiving a symbol expression for a location in the executable image file where the global breakpoint is to be placed;

passing the symbol expression and the file name to a first operating system module running on the computer system;

receiving a file offset corresponding to the symbol location from the first operating system module;

passing the file name for the executable image file to a second operating system module;

receiving a file identifier from the second operating system module, wherein the file identifier is used by the operating system for uniquely identifying the executable file in the computer system memory; and

representing said global breakpoint in code of said software using the~~an~~ received file identifier of the~~an~~ executable image file and the received ~~an~~ offset in said executable file.

2. (original) The method according to claim 1, wherein said file identifier is a file name.

3. (currently amended) The method according to claim 1, wherein said file identifier is an inode of a Unix ~~or Unix-like~~ operating system.

4. (original) The method according to claim 1, wherein said file identifier is a file control block of a non-Unix operating system.

5. (original) The method according to claim 1, further including the step of resolving a virtual address of said code to said file identifier and said offset..

6. (canceled)

7. (original) The method according to claim 1, further including the step of providing a hash list to look up said global breakpoint using said file identifier and said offset.

8. (original) The method according to claim 7, further including the step of maintaining said hash list of global breakpoints based on file identifiers and offsets.

9. (original) The method according to claim 1, further including the step of deriving said file identifier and said offset using a virtual address.

10. (original) The method according to claim 9, wherein said deriving step is dependent upon information maintained by an operating system to map executable files to memory.

11. (original) The method according to claim 9, further including the step of: determining file identifier and said offset from said virtual address for a memory mapped region.

12. (original) The method according to claim 9, wherein two or more virtual addresses exist for said software code.

13. (original) The method according to claim 1, wherein said global breakpoint is contained in a private-per-process copy of a physical page of said software code.

14. (currently amended) A computer-implemented apparatus for identifying a global breakpoint for debugging computer software, said apparatus including:

a central processing unit for executing said computer software;

memory for storing at least a portion of said computer software; ~~and~~

means for receiving a file name for an executable image file, wherein the executable image file is loaded in memory of a computer system and the global breakpoint is to be placed in the image for executing by the computer system;

means for receiving a symbol expression for a location in the executable image file where the global breakpoint is to be placed;

means for passing the symbol expression and the file name to a first operating system module running on the computer system;

means for receiving a file offset corresponding to the symbol location from the first operating system module;

means for passing the file name for the executable image file to a second operating system module;

means for receiving a file identifier from the second operating system module, wherein the file identifier is used by the operating system for uniquely identifying the executable file in the computer system memory; and

means for representing said global breakpoint in code of said computer software using the~~an~~ received file identifier of the~~an~~ executable image file and the received~~an~~ offset in said executable file.

15. (original) The apparatus according to claim 14, wherein said file identifier is a file name.

16. (currently amended) The apparatus according to claim 14, wherein said file identifier is an inode of a Unix ~~or Unix-like~~ operating system.

17. (original) The apparatus according to claim 14, wherein said file identifier is a file control block of a non-Unix operating system.

18. The apparatus according to claim 14, further including means for resolving a virtual address of said code to said file identifier and said offset.

19. (canceled)

20. (original) The apparatus according to claim 14, further including a hash list to look up said global breakpoint using said file identifier and said offset.

21. (original) The apparatus according to claim 20, further including means for maintaining said hash list of global breakpoints based on file identifiers and offsets.

22. (original) The apparatus according to claim 14, further including means for deriving said file identifier and said offset using a virtual address.

23. (original) The apparatus according to claim 22, wherein said deriving means is dependent upon information maintained by an operating system to map executable files to memory.

24. (original) The apparatus according to claim 22, further including means for determining said file identifier and said offset from said virtual memory address for a memory mapped region.

25. (original) The apparatus according to claim 22, wherein two or more virtual addresses exist for said software code.

26. (original) The apparatus according to claim 14, wherein said global breakpoint is contained in a private-per-process copy of a physical page of said software code.

27. (currently amended) A computer program product having a computer readable medium having a computer program recorded therein for identifying a global breakpoint for debugging computer software, said computer program product including:

computer program means for receiving a file name for an executable image file, wherein the executable image file is loaded in memory of a computer system and the global breakpoint is to be placed in the image for executing by the computer system;

computer program means for receiving a symbol expression for a location in the executable image file where the global breakpoint is to be placed;

computer program means for passing the symbol expression and the file name to a first operating system module running on the computer system;

computer program means for receiving a file offset corresponding to the symbol location from the first operating system module;

computer program means for passing the file name for the executable image file to a second operating system module;

computer program means for receiving a file identifier from the second operating system module, wherein the file identifier is used by the operating system for uniquely identifying the executable file in the computer system memory; and

computer program code means for representing said global breakpoint in code of said computer software using thean received file identifier of thean executable image file and the received ~~an~~ offset in said executable file.

28. (original) The computer program product according to claim 27, wherein said file identifier is a file name.

29. (currently amended) The computer program product according to claim 27, wherein said file identifier is an inode of a Unix ~~or Unix-like~~ operating system.

30. (original) The computer program product according to claim 27, wherein said file identifier is a file control block of a non-Unix operating system.

31. (original) The computer program product according to claim 27, further including computer program code means for resolving a virtual address of said code to said file identifier and said offset.

32. (canceled)

33. (original) The computer program product according to claim 27, further including computer program code means for providing a hash list to look up said global breakpoint using said file identifier and said offset.

34. (original) The computer program product according to claim 33, further including computer program code means for maintaining said hash list of global breakpoints based on file identifiers and offsets.

35. (original) The computer program product according to claim 27, further including computer program code means for deriving said file identifier and said offset using a virtual address.

36. (original) The computer program product according to claim 35, wherein said computer program code means for deriving is dependent upon information maintained by an operating system to map executable files to memory.

37. (original) The computer program product according to claim 35, further including computer program code means for determining said file identifier and said offset from said virtual memory address for a memory mapped region.

38. (original) The computer program product according to claim 35, wherein two or more virtual addresses exist for said software code.

39. (original) The computer program product according to claim 27, wherein said global breakpoint is contained in a private-per-process copy of a physical page of said software code.